A test methodology for interconnect structures of LUT-based FPGAs

Hiroyuki Michinishi* Tokumi Yokohira† Takuji Okamoto‡
Tomoo Inoue** Hideo Fujiwara††

*Okayama University
†Okayama University
‡Okayama University
**Nara Institute of Science and Technology
††Nara Institute of Science and Technology

A Test Methodology for Interconnect Structures of LUT-Based FPGAs

Hiroyuki Michinishi, Tokumi Yokohira, Takuji Okamoto
Department of Information Technology
Faculty of Engineering, Okayama University
3-1-1, Tsushima-naka, Okayama, 700 Japan

Tomoo Inoue, Hideo Fujiwara
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, 630-01 Japan

Abstract

In this paper, we consider testing for programmable interconnect structures of look-up table based FPGAs. The interconnect structure considered in the paper consists of interconnecting wires and programmable points (switches) to join them. As fault models, stuck-at faults of the wires, and extra-device faults and missing-device faults of the programmable points are considered. We heuristically derive test procedures for the faults and then show their validnesses and complexities.

1 Introduction

Field programmable gate arrays (FPGAs) are modern logic devices which can be programmed to implement logic circuits in the field[1,2]. There are many different architectures of FPGAs driven by different programming technologies. FPGAs with SRAM-based architecture, also called *look-up table based* FPGAs[1-3], are the most popular ones. Such types of FPGAs are called FPGAs briefly in this paper. FPGA consists of an array of identical configurable (programmable) logic blocks (CLBs), programmable I/O blocks (IOBs) and programmable interconnect structures. At present, FPGAs are widely used in rapid system prototypings and system reconfigurations, because of their reprogrammability.

Some researchers[4,5] have proposed testing for programmed FPGAs, in which logic circuits are implemented, but it is not applicable to test for unprogrammed FPGAs at the manufacturing time. On the other hand, we have considered testing for unprogrammed FPGAs and proposed simultaneous testing for all look-up tables (LUTs) of FPGAs under two different programming schemes; one for *sequential loading* programming scheme and the other for *random access loading* one[6]. This simultaneous testing ensures that all the corresponding programmed logic circuits on the FPGAs are fault-free, and the number of programs required for the testing is independent of the array size of the FPGAs. Testing for other components, however, have not been proposed there. In this paper, we present testing for programmable interconnect structures in the FPGAs with sequential loading.

Programmable interconnect structures are used to connect CLBs each other and consist of switch matrices (SMs) and programmable switch blocks (PSBs). In this paper, we focus on testing for PSBs. Each of them is divided into three segments, (i) wires between adjacent two SMs, (ii) wires connected to IOBs and switches to connect them to segment (i), and (iii) wires connected to CLBs and switches to connect them to segment (i).

In this paper, we show architecture of FPGAs, fault model of PSBs and test strategy. The strategy is to consider test procedures for the segments (i), (ii) and (iii) in this order, so that they are easily derived. We then derive the procedures and prove their validnesses.

2 FPGA architecture and interconnect structures

The architecture of FPGA is illustrated in Fig. 1. CLBs are circuits to implement logic functions, and PSBs ($PSB_1 \sim PSB_6$) and SMs are structures to connect CLBs each other. IOBs are circuits to connect the FPGA with external inputs/outputs. In addition to the circuits and the structures, the FPGA is equipped with a programming facility which is used to determine its function but not shown in Fig. 1. It is assumed in the succeeding discussions that the array size (the number of CLBs) of the FPGA is $N \times N$.

1081-7735/96 $5.00 © 1996 IEEE
Proceedings of ATS '96

Every CLB consists of one look-up table (LUT), two multiplexers (MUX1 and MUX2) and one D flip-flop (DFF) as shown in Fig. 2. r_1, r_2, $\cdots$, $r_{2^{3k}}$, r_{M1} and r_{M2} show SRAM cells, each of which is loaded with 0 or 1 at programming. When an input pattern $(x_1, x_2, \cdots, x_{3k})$ is applied to the LUT as an address, the content of the corresponding SRAM cell is read out as its output f. Thus, an arbitrary logic function of $3k$ variables can be realized. The output of MUX1 is f if $r_{M1} = 0$, otherwise the logical value on the wire y. Similarly, the logical value on the wire z is f if $r_{M2} = 0$, otherwise q (the output of the DFF).

PSBs are divided into six types, $\mathrm{PSB}_1 \sim \mathrm{PSB}_6$ as shown in Fig. 1. Fig. 3 shows four PSBs, $\mathrm{PSB}_1 \sim \mathrm{PSB}_4$, which surround the CLB located in the upper left corner of the FPGA. Each PSB has n wires to connect adjacent two SMs each other. In addition, PSB_1 and PSB_3 have k+4 and $2k$ vertical wires, respectively, and PSB_2 and PSB_4 have k+5 and k+2 horizontal wires, respectively. Small circles show switches (each of them is referred to as PS in this paper) to control connectivities of the corresponding cross points according to the contents of SRAM cells appended to them. Each PS joins the corresponding two wires each other if and only if the content of the corresponding SRAM cell is 1. Two triangles located in an IOB show three-state buffers which are controlled by the corresponding SRAM cells r_{Bi}'s $(i = 1, 2)$. Each of them is closed if and only if the content of the corresponding SRAM cell is 1. Each SM is a collection of switches which can connect every input wire with an arbitrary number of other wires, where any two wires from a PSB can not be connected. The connectivities of the switches are controlled by the contents of the corresponding SRAM cells (if and only if the content of an SRAM cell is 1, the corresponding switch connects the corresponding two wires each other).

We must load all the SRAM cells with logical values for the configuration of the FPGA. If we want to change a part of them after the configuration, we must reload all the SRAM cells.

Hereafter, for simplicity of the descriptions, a wire w between adjacent two SMs, a wire connecting w with a CLB and a wire connecting w with an IOB are referred to as $\mathrm{W_{PSB}}$, $\mathrm{W_{CLB}}$ and $\mathrm{W_{IOB}}$, respectively, and PSs on a $\mathrm{W_{CLB}}$ and a $\mathrm{W_{IOB}}$ are referred to as $\mathrm{S_{CLB}}$ and $\mathrm{S_{IOB}}$, respectively.

3 Fault model and test strategy

It is assumed in this paper that no circuits (including the programming facility) and no structures have any fault except at most one PSB which may have multiple faults. We consider stuck-at faults (SAFs) of wires, and extra-device faults (EDFs) and missing-device faults (MDFs) of PSs, where an EDF and an MDF are such faults that PSs are stuck at joint and disjoint states, respectively.

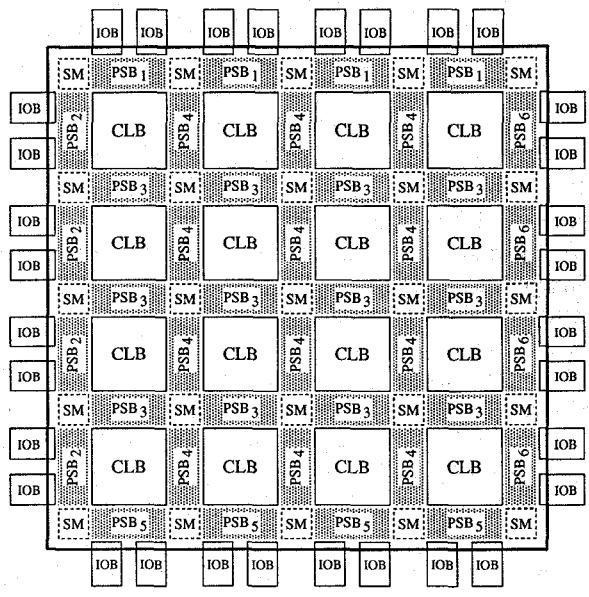


Figure 1. Architecture of FPGA

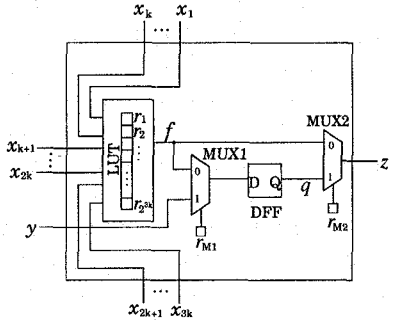


Figure 2. Structure of CLB

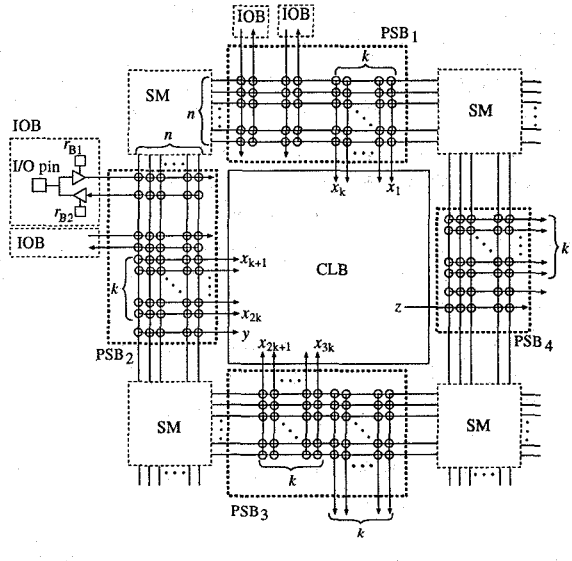


Figure 3. PSBs and SMs surrounding a CLB

For simplicity of the succeeding discussions, the following assumptions are introduced with respect to logical values on wires.

(A1) If a wire has any SAF, the logical value on it is invariant regardless of observed positions.

(A2) The logical value on an isolated wire is fixed at either 0 or 1, where an isolated wire is one in such a situation that it is floating from any drive source.

(A3) Assume that two wires are respectively driven from independent sources. Then, even if they are joined each other by a PS at joint state, logical values on them remain unchanged.

(A4) Assume that two wires l_1 and l_2 are respectively driven from independent sources. Then, even if they are joined each other through an isolated wire l_3 with two PSs at joint states, logical values on l_1 and l_2 remain unchanged and the logical value on l_3 is fixed at either 0 or 1.

Under the assumptions mentioned above, we will consider testing for W_{PSB}'s, W_{CLB}'s, W_{IOB}'s, S_{CLB}'s and S_{IOB}'s. The strategy is as follows. First, we will derive such a test procedure that if at least one W_{PSB} has any SAF, then some incorrect output appears independent of the presence of other faults. Second, under the condition that no W_{PSB} has any SAF, we will derive such a test procedure that if at least one of W_{IOB}'s and S_{IOB}'s has any fault, then some incorrect output appears independent of the presence of faults at W_{CLB}'s and S_{CLB}'s. Finally, under the same condition as the second procedure, we will derive such a test procedure that if at least one of W_{CLB}'s and S_{CLB}'s has any fault, then some incorrect output appears independent of the presence of faults at W_{IOB}'s and S_{IOB}'s. Using the three procedures, some incorrect output appears if any one of PSBs has at least one fault. Thus, we can attain 100 % fault coverage.

It is assumed in the succeeding sections that the contents of SRAM cells to which we do not refer are 0s, and any wire to which no test pattern is applied is isolated if it does not have any SAF and the logical value does not propagate to it.

4 Test procedure for W_{PSB}'s

Let v_j $(1 \leq j \leq n)$ be the j-th W_{PSB} from the left (top) in each of PSB_2's, PSB_4's and PSB_6's (PSB_1's, PSB_3's and PSB_5's) in Fig. 1.

We first consider SAFs at an arbitrary number of W_{PSB}'s in any one of PSB_2's, PSB_4's and PSB_6's. If there exist such faults, the following test procedure produces some incorrect output.

[TP-1 : Test procedure for W_{PSB}'s of PSB_2's, PSB_4's and PSB_6's] Execute (1) and (2) for $j = 1, 2, \cdots, n$ (see Fig. 4).

(1) Program so that (a) all v_j's of PSB_2's, PSB_4's and PSB_6's are joined through SMs as shown by a bold line in Fig. 4, (b) v_j of the PSB_2 located in the upper left corner is connected with a W_{IOB} (w_1) for the input and the corresponding three-state buffer in the corresponding IOB is closed, and (c) v_j of the PSB_6 located in the lower right corner is connected with a W_{IOB} (w_2) for the output and the corresponding three-state buffer is closed.

(2) Apply 0 and 1 to w_1 as two test patterns, and observe logical values on w_2 as the outputs. □

In Fig. 4, black PSs mean that they are programmed so as to be in joint states.

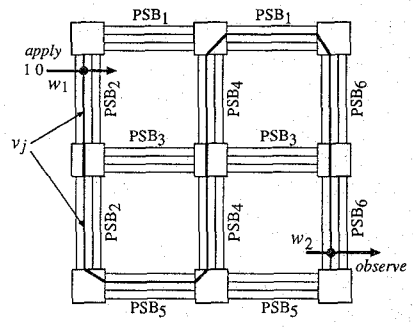


Figure 4. Program and test patterns for testing of W_{PSB}'s

Prior to the proof of the validness of TP-1, we introduce the following lemma.

[Lemma 1] Assume that there exists any SAF at any wire of W_{PSB}'s, W_{CLB}'s and W_{IOB}'s. Then, the logical value on it does not change even if the changes of logical values occur at an arbitrary number of wires except it in the FPGA. □

The proof is trivial from the assumptions (A1), (A3) and (A4). Using the lemma, the validness of TP-1 is proved as follows.

[Proof of the validness of TP-1] It is trivial that if the FPGA has no faults, the logical value on w_2 is always identical to that on w_1 in the procedure (2). On the other hand, if at least one of W_{PSB}'s in any one PSB has any SAF, the logical value on w_2 is as follows.

Assume that there exists any SAF at v_j of the PSB_2 located in the upper left corner. The logical value on the v_j is invariant from Lemma 1 independently of the logical values on w_1. On the other hand, since a single PSB fault model is introduced in this paper, all SMs and all PSBs except the PSB_2 have no faults. The logical value on the v_j therefore propagates to w_2. Thus, the logical value on w_2 is invariant.

Next, assume that there exists any SAF at v_j of the PSB_6 located in the lower right corner of the FPGA. (a) The logical value on the v_j is invariant from Lemma 1 regardless of the logical values on w_1. And (b)since a single PSB fault model is assumed, two SMs on the top and bottom of the

PSB_6 propagate logical values to no W_{PSB}'s except the v_j. From (a), (b) and the assumptions (A2) $\sim$ (A4), all W_{PSB}'s are unchanged. Therefore, the logical value on w_2 is unchanged.

Finally, if there exists any SAF at v_j of any one of PSBs except the PSB_2 and the PSB_6 mentioned above, the fixed logical value on the v_j propagates to w_2. □

Similarly, the test procedure for PSB_1's, PSB_3's and PSB_5's can be obtained by replacing PSB_2's, PSB_4's and PSB_6's in TP-1 with PSB_1's, PSB_3's and PSB_5's, respectively.

Thus, the number of programs required for testing of all PSBs is $2n$.

5 Test procedure for W_{IOB}'s and S_{IOB}'s

In this section, we consider testing for W_{IOB}'s and S_{IOB}'s under the assumption that no W_{PSB} has any SAF. First, we present a test procedure for a single PSB. Second, we extend it to a test procedure for all PSBs.

Fig. 5 illustrates layout of wires and PSs in a single PSB, where the symbols a_{ij}'s, w_i's and v_j's ($1 \leq i \leq 4$; $1 \leq j \leq n$) show S_{IOB}'s, W_{IOB}'s and W_{PSB}'s, respectively, and IO_l's ($1 \leq l \leq 2$) and A_j's show I/O terminals and boundary points between the PSB and the SM, respectively, and TB_i's show three-state buffers. If at least one of w_i's and a_{ij}'s has any fault, then the following procedure produces some incorrect output.

[TP-2 : Test procedure for W_{IOB}'s and S_{IOB}'s] (see Fig. 5)

(1) Execute (1-1) and (1-2) for $m = 1, 2$.
  (1-1) Program so that TB_{2m-1} is closed and any one (a_{2m-1j^*}) of a_{2m-1j}'s is in joint state.
  (1-2) Apply 0 and 1 to IO_m as two test patterns, and observe the outputs at A_{j^*}.

(2) Execute (2-1) and (2-2) for $m = 1, 2$.
  (2-1) Program so that TB_{2m} is closed and any one (a_{2mj^*}) of a_{2mj}'s is in joint state.
  (2-2) Apply 0 and 1 to A_{j^*} as two test patterns, and observe the outputs at IO_m.

(3) Execute (3-1) and (3-2) for $m = 1, 2$.
  (3-1) Program so that TB_{2m-1} is closed.
  (3-2) Apply 0 and 1 to IO_m as two test patterns, and observe the outputs at $A_1 \sim A_n$.

(4) Execute (4-1) and (4-2) for $m = 1, 2$.
  (4-1) Program so that TB_{2m} is closed.
  (4-2) Apply all 0s and all 1s to $A_1 \sim A_n$ as two test patterns, and observe the outputs at IO_m.

(5) Execute (5-1) and (5-2) for $j = 1, 2, \cdots, n$.
  (5-1) Program so that TB_1 and TB_4 are closed, and a_{1j} and a_{4j} are in joint states.
  (5-2) Apply 0 and 1 to IO_1 as two test patterns, and observe the outputs at IO_2.

(6) Execute the procedure which is obtained by replacing TB_1, TB_4, a_{1j}, a_{4j}, IO_1 and IO_2 in the procedure (5) with TB_3, TB_2, a_{3j}, a_{2j}, IO_2 and IO_1, respectively. □

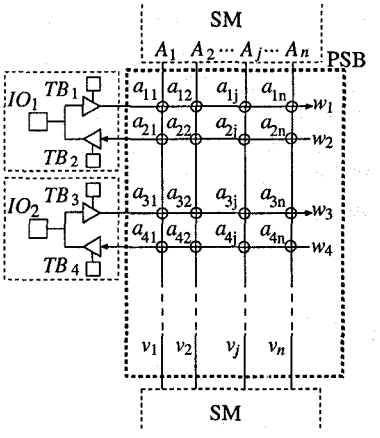


Figure 5. Layout of W_{PSB}'s, W_{IOB}'s and S_{IOB}'s in a single PSB

The validness of TP-2 can be proved as follows.

[Proof of the validness of TP-2] It is trivial that if the FPGA has no faults, the logical values appeared at the outputs (observation points) in the procedures (1-2), (2-2), (5-2) and (6-2) coincide with those of the corresponding test patterns, and two logical values at each output in each of the procedures (3-2) and (4-2) are equal each other. On the other hand, if at least one of w_i's and a_{ij}'s has any fault, the outputs are as follows.

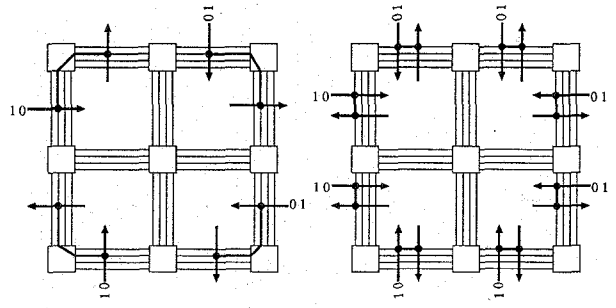
Assume that w_{2m-1} has any SAF for some m. Then, from Lemma 1, the logical values at IO_m can not propagate to w_{2m-1} in the procedure (1-2), consequently to A_{j*}. Therefore, the logical value at A_{j*} is unchanged independently of the logical value at IO_m. A similar discussion holds in the procedure (2-2). Thus, it can be assumed in the following discussions that no w_i ($1 \leq i \leq 4$) has any SAF.

Assume that a_{2m-1j} has an EDF for some m and some j. Then, the logical value at IO_m propagates to A_j in the procedure (3-2), since w_{2m-1} and v_j have no SAFs. Therefore, the logical value at A_j coincides with the corresponding test pattern. A similar discussion holds in the procedure (4-2). Thus, it is assumed in the following discussions that no a_{ij} ($1 \leq i \leq 4$; $1 \leq j \leq n$) has an EDF.

Assume that a_{1j} has an MDF for some j. Then, the logical value at IO_1 can not propagate to v_j in the procedure (5-2) of the j-th step, consequently to w_4. Thus, the logical value at IO_2 is kept unchanged. Similar discussions hold for a_{4j}, and for a_{3j} and a_{2j} in the procedure (6-2). □

Next, we consider the extension of TP-2 to a test pro-

cedure for simultaneous testing of several PSBs. Programming each couple of w_{2m-1} (for input; $m = 1, 2$) and w_{2m} (for output) through an SM as shown in Fig. 6(a), the procedures (1) and (2) can be simultaneously executed. Moreover, it is trivial that the procedure (5) (the procedure (6)) for each j can be executed for all PSBs simultaneously as shown in Fig. 6(b). Thus, in addition to executions of these procedures, if we execute the procedure (3) $4N$ times and the procedure (4) $4N$ times (note that $4N$ is the number of pairs of two IOBs in the FPGA), then testing of W_{IOB}'s and S_{IOB}'s in all PSBs can be attained. Therefore, the number of programs required for the testing is $16N + 2n + 4$.



(a) Simultaneous execution of the procedures (1) and (2)

(b) Simultaneous execution of the procedures (5)

Figure 6. Scheme for simultaneous testing of W_{IOB}'s and S_{IOB}'s

6 Test procedure for W_{CLB}'s and S_{CLB}'s

For simplicity, we refer to W_{CLB}'s for inputs of LUTs and for inputs of MUX1s and outputs of MUX2 (see Fig. 2) as W_{LUT}'s and W_{yz}'s, respectively, and refer to S_{CLB}'s on W_{LUT}'s and W_{yz}'s as S_{LUT}'s and S_{yz}'s, respectively.

We first present a test procedure for W_{LUT}'s and S_{LUT}'s under the assumption that no W_{PSB} has any SAF. Fig. 7 illustrates layout of wires and PSs in three PSBs which surround a CLB, where the symbols b_{ij}'s, h_i's and v_j's ($1 \leq i \leq k$; $1 \leq j \leq n$) show S_{LUT}'s, W_{LUT}'s and W_{PSB}'s, respectively, and A_j's and B_j's show boundary points between PSBs and SMs. If at least one of h_i's and b_{ij}'s in any one of the three PSBs has any fault, then the following test procedure produces some incorrect output.

[TP-3 : Test procedure for W_{LUT}'s and S_{LUT}'s] Execute (1) and (2) for $j = 1, 2, \cdots, n$ (see Fig. 7).

(1) Execute (1-1) and (1-2).

(1-1) Program so that (a) all b_{ij}'s ($1 \leq i \leq k$) of the three PSBs are in joint states, (b) all v_{j*}'s ($1 \leq j^* \leq n$) of the three PSBs are connected through SMs as shown in Fig. 7, (c) the contents of SRAM cells r_1 and $r_2 \sim r_{2^{3k}}$ are 0 and 1s, respectively.

(1-2) Apply 0 to A_j as the test pattern, and observe the logical value on z and the logical value at each B_{j*} ($1 \leq j^* \leq n$).

(2) Execute (2-1) and (2-2).

(2-1) Program so that (a) and (b) in the procedure (1-1) are attained, and the contents of SRAM cells $r_1 \sim r_{2^{3k}-1}$ and $r_{2^{3k}}$ are 1s and 0, respectively.

(2-2) Apply 1 to A_j as the test pattern, and observe the logical value on z and the logical value at each B_{j*} ($1 \leq j^* \leq n$).

The validness of TP-3 can be proved as follows.

[Proof of the validness of TP-3]

If the FPGA has no faults, then in the j-th step for each j, the logical values on z in the procedures (1-2) and (2-2) are 0s and the logical values at each B_{j*} ($1 \leq j^* \leq n$; $j^* \neq j$) in them coincide with each other. On the other hand, if any one of the three PSBs has any fault, the logical values on z and the logical values at each B_{j*} ($1 \leq j^* \leq n$) are as follows.

Without loss of generality, we assume that the PSB on the left of the CLB may have any fault. Assume that h_{i*} of the PSB has any SAF for some i^*, and let h^1_{ij} and h^2_{ij} be the logical values on each h_i in the procedures (1-2) and (2-2) of the j-th step, respectively. Then, from Lemma 1, $h^1_{i^*j} = h^2_{i^*j}$ for an arbitrary j. On the other hand, since a single PSB fault model is introduced, the CLB (the LUT) has no faults.

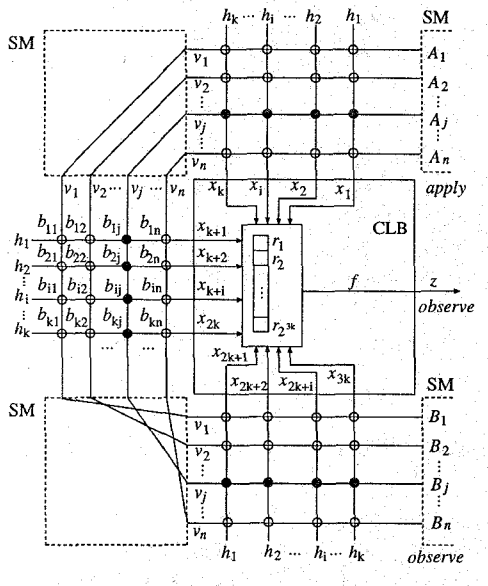


Figure 7. Test scheme for W_{LUT}'s and S_{LUT}'s in three PSBs

Therefore, either the logical value on z in the procedure (1-2) or that in (2-2) is 1. Consequently, an incorrect output appears on z. Thus, it can be assumed in the succeeding discussions that no h_i $(1 \leq i \leq k)$ has any SAF.

Assume that $b_{i^*j^*}$ has an EDF for some i^* and some j^*. If an incorrect output is observed on z in the procedure (1-2) or (2-2) of the j-th step for some j, then we can conclude that the FPGA has some EDFs and/or MDFs. If a correct output is observed on z for an arbitrary j, at least one of the logical values at $B_1 \sim B_n$ is incorrect as follows. If $h^1_{ij} = h^2_{ij}$ for some i and some j, then an incorrect output appears on z in the j-th step. Therefore, we can consider that $h^1_{ij} \neq h^2_{ij}$ for an arbitrary i and an arbitrary j. On the other hand, since it is assumed that v_{j^*} does not have any SAF, $h^1_{i^*j}$ and $h^2_{i^*j}$ propagate to B_{j^*} for each j. Thus, an incorrect output appears at B_{j^*} in the j-th step $(j \neq j^*)$. It can be therefore assumed in the succeeding discussions that no b_{ij} $(1 \leq i \leq k;\ 1 \leq j \leq n)$ has an EDF.

Finally, assume that $b_{i^*j^*}$ has an MDF for some i^* and some j^*. Since h_{i^*} does not have any SAF and no b_{i^*j} $(1 \leq j \leq n; j \neq j^*)$ has an EDF, h_{i^*} is isolated in the j^*-th step. From the argument and the assumption (A2), $h^1_{i^*j^*} = h^2_{i^*j^*}$. Thus, an incorrect output appears on z. □

As shown in Fig. 8, TP-3 can be extended to a test procedure for simultaneous testing of PSBs whose W_{LUT}'s are connected to CLBs in a column of the FPGA. Therefore, the number of programs required for testing of all W_{LUT}'s and all S_{LUT}'s is $N \times 2n$, since the number of programs in TP-3 is $2n$.

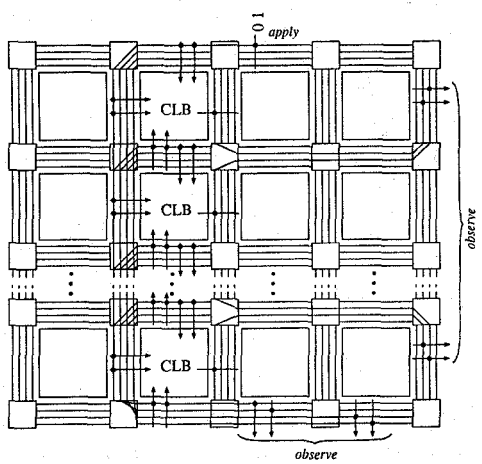


Figure 8. Program and test patterns for simultaneous testing of several PSBs

Finally, we consider testing for W_{yz}'s and S_{yz}'s. In each PSB, let c_j and d_j $(1 \leq j \leq n)$ be PSs on an input wire y of a CLB and an output wire z of a CLB, respectively. In Fig. 1, let CLB^t_s be the CLB located in the s-th row and t-th column of the FPGA $(1 \leq s \leq N; 1 \leq t \leq N)$, and let PSB^t_s and PSB^{N+1}_s $(1 \leq s \leq N; 1 \leq t \leq N)$ be the PSBs located on the left of CLB^t_s and the right of CLB^N_s, respectively. If at least one of W_{yz}'s and S_{yz}'s in any one of PSBs has any fault, the following procedure produces some incorrect output.

[TP-4 : Test procedure for W_{yz}'s and S_{yz}'s]

(1) Execute (1-1) and (1-2) for $j = 1, 2$ in each t $(1 \leq t \leq N)$ (see Fig. 9).

(1-1) Program so that (a) the contents of both r_{M1} and r_{M2} (see Fig. 2) in each $CLB^{t^*}_s$ $(1 \leq s \leq N; 1 \leq t^* \leq t)$ are 1s, (b) c_j in each $PSB^{t^*}_s$ $(1 \leq s \leq N;\ 1 \leq t^* \leq t)$ is in joint state, (c) d_j in each $PSB^{t^*}_s$ $(1 \leq s \leq N; 2 \leq t^* \leq t+1)$ is in joint state, (d) v_{j^*} in PSB^t_s and v_{j^*} in PSB^t_{s+1} $(1 \leq s \leq N-1; 1 \leq j^* \leq n;\ j^* \neq j)$ are connected each other through an SM, (e) the logical value on a W_{IOB} (α_s) propagates to y in each PSB^1_s $(1 \leq s \leq N)$, (f) the logical value on z in each CLB^t_s $(1 \leq s \leq N)$ propagates to a W_{IOB} (β_s), and (g) the logical value on each v_{j^*} $(1 \leq j^* \leq n; j^* \neq j)$ in PSB^t_N propagates to a W_{IOB} (γ_{j^*}).

(1-2) Apply all 0s and all 1s to $\alpha_1 \sim \alpha_N$ as two test patterns, and observe the logical values on $\beta_1 \sim \beta_N$ and all γ_{j^*}'s $(1 \leq j^* \leq n; j^* \neq j)$ as the outputs.

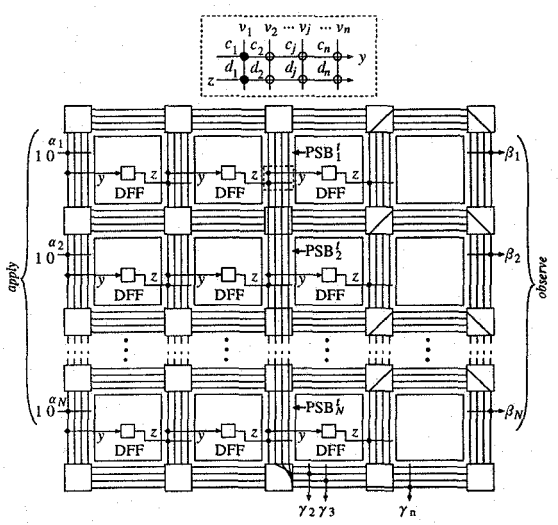


Figure 9. Program and test patterns for testing of SAFs and EDFs in PSB^t_s's $(1 \leq s \leq N;\ 1 \leq t \leq N)$

(2) Execute (2-1) and (2-2) for $j = 1, 2$ (see Fig. 10).

(2-1) Program so that (a) the contents of both r_{M1} and r_{M2} in each CLB are 1s, (b) all c_j's and all d_j's are in joint states, (c) v_{j^*} in PSB^{N+1}_s and v_{j^*} in PSB^{N+1}_{s+1} $(1 \leq s \leq N-1; 1 \leq j^* \leq n;\ j^* \neq j)$ are connected

each other through an SM, (d) the logical value on a W_{IOB} (α_s) propagates to y in each PSB_s^1 ($1 \leq s \leq N$), (e) the logical value on z in each CLB_s^N ($1 \leq s \leq N$) propagates to a W_{IOB} (β_s), and (f) the logical value on each v_{j^*} ($1 \leq j^* \leq n$; $j^* \neq j$) in PSB_N^{N+1} propagates to a W_{IOB} (γ_{j^*}).

(2-2) execute the same procedure as (1-2).

(3) Execute (3-1) and (3-2) for $j = 1, 2, \cdots, n$ (see Fig. 11).

(3-1) Program so that (a), (b), (d) and (e) in the procedures (2-1) are attained.

(3-2) Apply all 0s and all 1s to $\alpha_1 \sim \alpha_N$ as two test patterns, and observe the logical values on $\beta_1 \sim \beta_N$ as the outputs. □

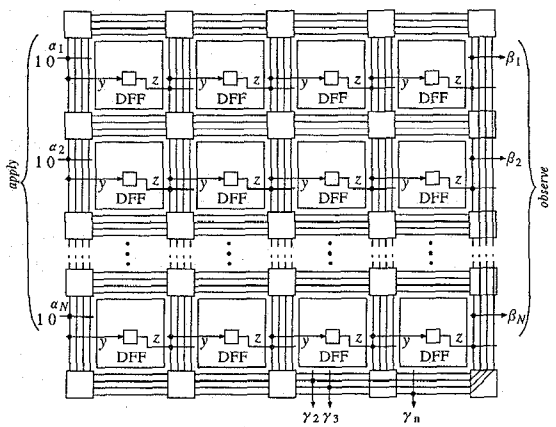


Figure 10. Program and test patterns for testing of SAFs and EDFs in PSB_s^{N+1}'s ($1 \leq s \leq N$)

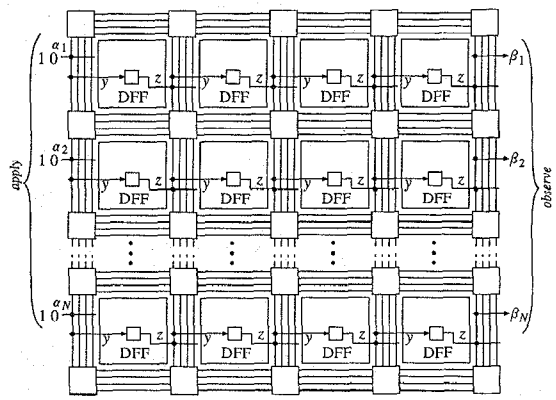


Figure 11. Program and test patterns for testing of MDFs in PSB_s^t's ($1 \leq s \leq N; 1 \leq t \leq N+1$)

The procedure (1) produces some incorrect output if there exists at least one fault of SAFs and EDFs in any one of PSB_s^t's ($1 \leq s \leq N; 1 \leq t \leq N$), and similarly, the procedure (2) produces some incorrect output if there exists at least one such fault in any one of PSB_s^{N+1}'s ($1 \leq s \leq N$). Under the assumption that no PSB has any fault except MDFs, the procedure (3) produces some incorrect output if there exists at least one MDF in any one of all PSBs of the FPGA. The proof of the validness of TP-4 is omitted due to space limitation. The number of programs in TP-4 is $2N + n + 2$.

7 Conclusion

In this paper, we considered testing for PSBs of FPGAs, such that it ensures that all the programmed PSBs are fault-free. And we heuristically derived test procedures for PSBs in which the number of programs required to test all PSBs of the FPGA is $2Nn+18N+5n+6$. When they are applied to FPGAs, the time T required to test all PSBs is nearly equal to the time required to load all the programs (the time required to apply the test patterns is negligible small compared with the loading time). For example, in the case of XC2064[3]($N = 8, n = 4$, $t_c = 100$ ms, where t_c is the time required to load each program), T is about 23.4 seconds.

It is one of our works to consider testing for switch matrices and I/O blocks. It is also an important work to obtain an efficient testing for the whole of FPGAs, by integrating the test procedures for all components of FPGAs.

References

[1] S. D. Brown, R. J. Francis, J. Rose and Z. G. Vranesic, "Field-programmable gate arrays," Kluwer Academic Publishers, 1992.

[2] S. M. Trimberger, "Field-programmable gate array technology," Kluwer Academic Publishers, 1994.

[3] *The Programmable Logic Data Book*, Xilinx Inc., 1994.

[4] I. Pomeranz and S. M .Reddy, "Testability considerations in technology mapping," *Proc. ATS '94*, pp. 151–156, Nov. 1994.

[5] H. Tsuboi, H.Nakada and T. Miyazaki, "Testing for circuits realized as FPGAs using register insertion method," Technical report of IEICE, *FTS94-53*, pp. 55–60, Oct. 1994.

[6] T. Inoue, H. Fujiwara, H. Michinishi, T. Yokohira and T. Okamoto, "Universal Test Complexity of Field-Programmable Gate Arrays," *Proc. ATS '95*, pp. 259-265, Nov. 1995.